THE PLANNING COORDINATOR (PCOORD) FOR ROBUST ERROR RECOVERY AND DYNAMIC ON-LINE PLANNING OF ROBOTIC TASKS

NAGW-1333

by

J.J. Farah and R.B. Kelley

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering Department
Troy, New York 12180-3590

December 1991

CIRSSE REPORT #107

ABSTRACT

The Planning Coordinator is a logical extension to the Coordination Level of the Intelligent Machine Model, functioning to provide a heretofore unavailable platform for robust error recovery and dynamic on-line planning by autonomous and semi-autonomous robotic systems.

This paper introduces the Planning Coordinator and focuses upon its macro-structure, interfaces and functional description, with respect to its role as the mechanism whereby an existing robotic system requiring significant human intervention can be made more autonomous, thus becoming more robust.

1.0 Introduction

Preliminary research is presented that stems from the need for robust error recovery and a base mechanism from which to engage in the dynamic on-line planning of robotic tasks by autonomous robotic systems. Called the Planning Coordinator (PCOORD) [1], the device which achieves the above, is a physically distinct logical extension to the Coordination Level of the Intelligent Machine Model developed by Saridis [2-4].

The Intelligent Machine Model introduces a three level description of the activities of an intelligent machine based upon the concept of Increasing Precision with Decreasing Intelligence. **Figure 1** depicts the three levels of the Intelligent Machine Model with the Organization Level, representing the highest degree of intelligence, at the apex and the Execution Level, representing the highest degree of precision, at the base.

The Coordination Level is an intermediate level between the Organization and Execution levels that functions to provide among other things a suitable interface between the two extreme levels. This paper focuses on the Planning Coordinator shown in **Figure 2**. The Planning Coordinator

provides robust error recovery, a suitable platform from which to enact on-line planning, and a minimal backup mechanism in the case of catastrophic failure of the Intelligent Machine. The Macro-Structure of the Planning Coordinator as well as its functional description within the Intelligent Machine Hierarchy is introduced. Both the Macro-Structure and the functional description are presented from a high-level perspective with a limited amount of implementation detail, the particulars of which are continuing through refinements and additional research.

Previous work in the development of robust error recovery mechanisms has been attempted by DiCesare, Fielding, and Goldbogen [5] utilizing Petri Nets. In addition, significant amounts of work in the development of functional robustness within the Intelligent Machine Model have been performed by Kelley and Moed [6], but not in terms of robust error recovery and on-line planning of robotic tasks.

The document is organized into the following sections:

2.0 Planning Coordinator (PCOORD): Macro-Structure

Physically, the Planning Coordinator is a stand alone device that functions as a logical extension to the Intelligent Machine. As shown in **Figure 3**, it is designed to function as one of the Coordinators in the Coordination Level of the Intelligent Machine Hierarchy. However, the design is intended to be flexible enough that the Planning Coordinator can be adapted to function as an Intelligent Machine with a more limited capacity than a full Intelligent Machine Model implementation.

The block diagram of **Figure 3** shows the functional interconnections between various aspects of the Intelligent Machine. The Planning Coordinator is interconnected to and logically located with other coordinators in the Coordination Level. Unlike the other coordinators, however, the Planning Coordinator does not directly communicate with the Organization or Execution Levels of the Intelligent Machine Model. Instead, when communication to either level is necessary, the Planning Coordinator communicates through either the Dispatcher or other coordinators.

As a logical extension to the Intelligent Machine, the

Planning Coordinator is logically subservient to the Main Controller of the Intelligent Machine. This structure is maintained at all times except in the event of a catastrophic failure of the Intelligent Machine's Main Controller. Only then does the PCOORD become system master, here in an error recovery role.

The block diagram of the Planning Coordinator is expanded in **Figure 4** to introduce its constituent parts. These parts accomplish the primary functions delegated to the Planning Coordinator. The constituent parts are listed below:

Current World Model (CWM)
Shadow-Coordination Level Petri Net (SCPN)
Primitive Structure Database (PSDB)
Mapping Mechanism (MM)
Node/Link Weighting Mechanism (N/L-WM)
Error Recovery Generation Algorithm (ERGA)
System Fault Monitor (SFM)

With the exception of the System Fault Monitor these functions constitute the mechanisms whereby a robotic task error recovery or on-line plan is successfully enacted. The System Fault Monitor provides a mechanism to determine the state of the computer systems involved in the robotic operation and has nothing to do with the execution, planning or recovery of robotic tasks. However, since it performs an error test it is included in the Planning Coordinator.

Given below are the definitions and descriptions of the parts constituting the Planning Coordinator:

Current World Model:

The Current World Model is a dynamically changing representation of the environmental information available to the Intelligent Machine. Its function is to accurately represent the present status of the Intelligent Machine's environment at all times. This requires the continuous updating of the Current World Model as the Intelligent Machine executes tasks which change its environment.

At any one time, only a small portion of the Current World Model will be updated. This is similar to the "locality principle" utilized in Computer Engineering.

Since the amount of message traffic on the Intelligent Machine's communication buses will in general be high, especially during an error recovery or on-line planning sequence, utilizing the Current World Model, minimizes the overall message traffic. In turn, this limits communication bus loading.

NOTE: Not a part of the Planning Coordinator is a static world model called a Global World Model (GWM). It is a static representation of the initial environmental information available to the Intelligent Machine. Hence, it functions to provide a history for the robotic system. Its maintenance is the responsibility of the Organization Level of the Intelligent Machine.

Shadow-Coordination Level Petri Net:

The Shadow-Coordination Level Petri Net is an exact copy of the executing Coordination Level Petri Net generated via the Dispatcher. It functions to follow (shadow) the execution of the Coordination Level Petri Net.

This function is necessary so that when an error recovery or on-line plan is required, the exact location at which the error recovery or on-line plan is to take place, is known immediately.

Knowing the location of the error recovery or on-line plan significantly decreases the overall execution time of the error recovery or on-line plan, since error recovery routines based upon the most likely errored event can be pre-calculated and linked to SCPN transitions for immediate execution. The same is true for some on-line plans as will be described in Section 4.

NOTE: It is possible to pre-calculate several alternative error recovery or on-line paths based upon knowledge of the robotic system. These paths each possess a likelihood of success which would permit them to be rank ordered for execution from most likely to succeed to least likely to succeed. Paths which are executed and succeed are rewarded the 'next time around' with a higher likelihood rating. Similarly paths that do not succeed are penalized with a lower likelihood of success rating.

All of the error recovery and on-line plan paths are created from primitive structures that reside in the Primitive Structure Database.

Primitive Structure Database:

The Primitive Structure Database is a relational database composed of Primitive Structures.

A Primitive Structure (PS) is defined as: a simple, basic building block created by the Planning Coordinator, based upon environmental information contained in the Current World Model. Several are combined to form more complex structures that are later used for robust error recovery and/or on-line planning.

The PSs used are themselves live, safe, bounded reversible Petri Nets and utilize the synthesis techniques developed in [7-10] to form more

complex structures for more involved robotic tasks.

For robotic activities, such primitive structures would include: Grasp, Rotate, Move_Link, Un_Grasp, Insert, Remove, etc.

The primitive structure database is designed to be a relational database so that the Planning Coordinator can make use of the stored primitives, insert new primitives into the database and/or delete existing primitives from the database.

Since the primitive structures in the database may share characteristics, and the speed of insertion, deletion and membership is of paramount importance in the real-time operation of the intelligent machine, the implementation of both the primitive structures and the primitive structure database reflects those needs.

The means of facilitating the insertion and retrieval of data to/from the database is to provide a one-to-one mapping mechanism.

Mapping Mechanism:

The desired one-to-one Mapping Mechanism is a methodology to efficiently perform the following three functions:

1) Connect Shadow-Coordination Level Petri Net transitions to Map Interface Nodes which function as the start and end points of an error recovery or on-line plan sequence.

2) Connect PS interface nodes to their corresponding primitive structures in the primitive structure database.

3) Create Experience Vector Nodes based upon previously executed successful error recoveries.

Robustness requires that the mapping mechanism

be able to provide a fully interconnected network for the three types of 'nodes' mentioned. This network thus provides an interconnecting link from one node to all other nodes regardless of type.

Utilizing full interconnectivity permits large combinations of connections. The increase in the number of possible combinations introduces a flexibility not available in non-fully connected networks. The choice of which connections and nodes to assert is determined by the Node/Link Weighting Mechanism.

Node/Link Weighting Mechanism:

The Node/Link Weighting Mechanism assigns weights to the Nodes and Links that comprise an error recovery or on-line plan path.

The weights assigned to the Links are the likelihood of success ratings described previously. All Links between Nodes on the most likely to succeed path are given the same weight.

The weights assigned to Nodes indicate the order of execution of the primitive structures mapped to those Nodes, for that execution path. Hence it is possible that a Node may be executed first in one path, but fourth in another.

As described earlier, several possible paths may be available, based upon the occurrence of a particular error. The Link weights are used to represent the order of execution of these differing paths. If one path is successful the others are discarded.

NOTE: To be active, a Link weight must exceed a certain threshold value. This threshold functions to limit the possible number of paths to choose from for any given error (i.e. a path with a likelihood of success rating of .0001 is probably unlikely to succeed).

The process of assigning Link weights occurs in three steps:

1) Identify Nodes to be used for error recovery or on-line planning.

2) Weight representative Nodes and Links.

3) Establish the threshold to preen the number of possible alternate paths.

Steps 1 through 3 can be accomplished while the Planning Coordinator is not actively pursuing an error recovery or on-line plan. This permits a pre-calculation of the most likely errored event from the data available in the CWM before one is actually needed. When the Error Recovery Generation Algorithm (ERGA) is needed, it may not need to generate a new plan, if the error corresponds to the precalculated, most likely error event. This saves a significant amount of execution time.

If an error is not the most likely error event, then the ERGA must generate another error recovery plan. This is described below in the description of the Error Recovery Generation Algorithm.

Error Recovery Generation Algorithm:

The Error Recovery Generation Algorithm is the mechanism whereby error recoveries and on-line plans, as discussed in Section 4, are initiated and executed. The steps involved in the error recovery generation are outlined below. Note that the steps assume a certain amount of information is available from the Intelligent Machine. This information includes:

1) Error Status based upon a flag called ERR_FLAG. If asserted, an error is present. If not, no error is present.

2) On-Line Plan Status based upon a flag called OP_FLAG. If asserted alone, a type of on-line plan is desired.

3) Intelligent Machine Main Controller Status based upon a flag called MC_FAIL. If asserted, the Intelligent Machine Main Controller has failed.

The ERGA implements the following steps, a block diagram of which is given in **Figure 5**.

Step 1: Identify Error: Utilizing ERR_FLAG, OP_FLAG and MC_FAIL, identify the state of the error. For ERR_FLAG & OP_FLAG, determine the transition from which the error resulted and classify the error type if possible. For MC_FAIL, assume system control, and attempt system restoration. If successful, return system control to Intelligent Machine Main Controller. If unsuccessful, notify human operator base for assistance and place the Intelligent Machine in a quiescent state.

NOTE: There are several types of errors possible in the execution of a robotic task, from minor tolerance errors to irrecoverable errors.

Step 2: Notify Intelligent Machine Main Controller: Notify the Intelligent Machine's Main Controller that an error has occurred and activate the Map interface node corresponding to the errant SCPN transition.

Step 3: Control Request: If the error information from Step 1 is insufficient to begin error recovery generation, request control of the Intelligent Machine from the Main Controller.

If granted, system control is obtained to prevent other system operations from interfering with the error recovery.

If denied, re-try four times. If denied all four times abort error recovery or on-line plan.

Step 4: Analyze Error: Utilizing error information gathered in Step 3, the information provided via the ERR_FLAG and OP_FLAG, and environmental information from the CWM, determine the type of the error.

Step 5: Recovery Node/Link Activation: Based upon the results of Step 4, use the Node/Link Weighting Mechanism to activate the nodes and links required for the error recovery or on-line plan.

Step 6: Establish Primary & Secondary Error Recovery Route(s): Connect the primary error recovery or on-line plan path to the asserted Map Interface Node. Prepare secondary path(s) for execution, if needed.

Step 7: Primary Route Execution: Execute the primary error recovery route. If successful, update the

Global World Model, if requested. Return operation and control to the Intelligent Machine Main Controller, and terminate.

If the error recovery is unsuccessful, proceed to Step 8.

NOTE: During execution, the CWM is constantly being updated. Hence no special request for update of the CWM is needed. However, the GWM maintains a static base history of the robotic system. To update it indicates a basic problem with the initial environmental information provided to the Intelligent Machine. Hence to update it is the choice of the Intelligent Machine Main Controller.

Step 8: Secondary Execution: Execute Secondary recovery route(s). If successful, adjust weighting of links and nodes to reflect successful paths. Update the GWM if requested to and return control to the Intelligent Machine Main Controller.

If unsuccessful, repeat Step 8 for the next ordered execution path and continue until either success, or the set of possible paths is exhausted. If successful continue as in Step 7. If the set of possible paths is exhausted, notify the Intelligent Machine Main Controller of an irrecoverable failure and release control to the Intelligent Machine.

This completes the description of the Macro-Structure of the Planning Coordinator. The following section identifies the interfaces that are required for the functional operation of the Planning Coordinator. It is followed by a functional description of the Planning Coordinator.

3.0 Planning Coordinator: Interface Description

The requirements of the operation of the Planning Coordinator

neccesitate significant communication between it and the remainder of the Intelligent Machine. Based upon **Figure 3**, the Planning Coordinator requires access to the following Coordination Level subsystems:

Dispatcher
Sensor Coordinator
Arm Coordinator
Vision Coordinator

Since the intent of the Planning Coordinator is to make use of these subsystems' functions, a common high level communication scheme is needed to communicate between the Planning Coordinator and the other Coordinators. This communication scheme must necessarily accommodate high volume, high speed, possibly asynchronous message traffic. A single-bus, packet communication scheme can handle the volume and speed of messages while also providing error-correction capabilities.

Connection to the Organization Level is another interface needed by the Planning Coordinator, to relay error information and update information to the Intelligent Machine Main Controller and the Global World Model. This interface is accomplished through the Dispatcher. In addition, connection to the Execution Level, is accomplished through the existing Coordinator interfaces.

One possible architecture to facilitate the above is a single-bus, multi-point addressing scheme. This facilitates packet communication, while tending not to load down the communication bus with unnecessary traffic. **Figure 6** shows such an architecture. A dual bus architecture, as depicted in **Figure 7**, is preferred to permit high speed communication between the Planning Coordinator and the other Coordinators during an error-recovery or on-line plan over a second, dedicated, higher-bandwidth communication bus. This second bus could also be used as a monitor or excess capacity bus at other times.

The Planning Coordinator requires certain internal interfaces to maintain its overall speed of operation. Because the Primitive Structure Database and Current World Model require large amounts of storage, and necessitate high volume data transactions, a high speed, high volume communication bus between the Mapping Mechanism and the Current World Model, and the Mapping Mechanism and the Primitive Structure Database is needed. Lower speed communication buses between the SCPN and the Mapping Mechanism, and between the Mapping Mechanism, the Node/Link Weighting Mechanism, and the ERGA module provide sufficient communication bandwidth for them to function efficiently. The Planning Coordinator's internal architecture is depicted in **Figure 8**.

This concludes the description of the high-level interfaces required by the Planning Coordinator. The following section provides a functional description of the operation of the Planning Coordinator.

4.0 Planning Coordinator: Functional Description

The Planning Coordinator (PCOORD) is designed to be a device which can be configured to work in conjunction with the Intelligent Machine Model at the Coordination Level. It is a logical extension to the Coordination Level of the Intelligent Machine, although it is not physically part of it. This is an important distinction as is explained in the following subsection.

4.1 Functionality of the Planning Coordinator

The primary functions of the Planning Coordinator are to efficiently enact dynamic, robust error recovery within the logical hierarchy of the Intelligent Machine Model, as well as to act as a minimal backup to the Intelligent Machine in the event of the catastrophic failure of the Intelligent Machine's Main Controller. During such an occurrence the Planning Coordinator notifies some human operator or central

communication center of irrecoverable Intelligent Machine failure beyond its capabilities to repair, and places the Intelligent Machine in a quiescent state. Note that the catastrophic failure of the Intelligent Machine's Main Controller is not to be confused with an irrecoverable error generated by a robotic task. An irrecoverable error of this type is the result of either the Planning Coordinator's inability to provide a suitable error recovery plan or the Intelligent Machine is incapable of affecting the plan. The choice to notify some external (to the Intelligent Machine Robotic Platform) source is then the responsibility of the Main Controller of the Intelligent Machine, not the Planning Coordinator.

A secondary function of the Planning Coordinator, viewed as an extension to the error recovery platform, is a dynamic on-line planner. An on-line planning sequence is a particular instantiation of an error recovery. Including the on-line planner into the error recovery platform is a natural extension.

The primary and secondary functions of the Planning Coordinator give rise to two modes of operation: Error Recovery and On-Line Planning. These two modes of operation are the subjects of the next two subsections.

4.2 Error Recovery

Error recovery is the default mode of operation for the Planning Coordinator. In this mode the Planning Coordinator is in either an observer or an actor state. The Planning Coordinator's default state is observer, monitoring the execution of the Intelligent Machine Coordination Level Petri Net via the SCPN described in Section 2. While in this state the Planning Coordinator is a strict monitor and does not interfere in the operation of the Intelligent Machine. The Planning Coordinator changes to the actor state only when an error occurs. Then it takes an active role in the execution of robotic tasks only to provide error recovery plans.

Note that as mentioned in Section 2, there are several different types of errors that can result from an action. These different types range from simple tolerance errors to irrecoverable errors, each requiring different levels of interaction between the Planning Coordinator and the remainder of the Intelligent Machine.

The operation of the PCOORD while in error recovery mode is given in the block diagram description of **Figure 9**. Each of the main blocks, B1-B4, is described below:

B1: The PCOORD monitors the action of the SCPN and waits for an interrupt representing the assertion of the ERR_FLAG, OP_FLAG, or MC_FAIL flag described in Section 2. While not asserted, the PCOORD may d other things, for example clean-up the primitive structure database or create the most likely errored event recovery route.

When however, the ERR_FLAG or MC_FAIL flag is asserted the PCOORD changes from an observer to an actor and the error recovery is initiated.

B2: Once the appropriate flag is set and the PCOORD state is changed from observer to actor, the PCOORD begins its error recovery by first requesting error information from the intelligent machine. If the information is available, the PCOORD reacts to it and creates the appropriate recovery route utilizing the ERGA and available information.

The PCOORD then requests control of the Intelligent Machine from the Intelligent Machine's Main Controlle and, if granted, executes the error recovery plan. I denied, the PCOORD requests control four more times. If all are denied, the error recovery is aborted.

If the error information is not available or is insufficient, the PCOORD immediately requests control of the Intelligent Machine from the Intelligen Machine's Main Controller. If granted, the PCOOR attempts a Failure Mode Analysis of the 'error.'

If not granted, the PCOORD requests four more times. If all are denied, the error recovery is aborted.

NOTE: Since the PCOORD utilizes the SCPN, it always has a starting point from which to begin the error recovery. This cuts down on the search space the PCOORD would otherwise have to examine to fully characterize the error.

Once the Failure Mode Analysis is complete, the PCOORD determines a course of action to take. When the plan is complete, the PCOORD asks the Intelligent Machine Main Controller for clearance to execute the error recovery plan. If granted, the PCOORD executes the error recovery.

NOTE: The PCOORD is subservient to the Intelligent Machine's Main Controller which can abort an error recovery at any time. The request for clearance is made since it is likely that an error recovery will take a substantial amount of time to complete, depending on the error, at which time the resources available to the Intelligent Machine for other non-robotic task related work are minimal.

B3: The Error Recovery Generation Algorithm has been utilized to generate a plan of action. The steps involved in the execution of the plan are immediate and occur as follows.

Based upon the weighting and ordering of the PS Interface Nodes and their connection to Map Interface Nodes, each component primitive in the recovery path is executed and monitored for error. If a new error is caused during the error recovery, it is handled in the same manner as any other, with the difference being that the new error is handled first. Requests for system control for these new, nested errors are made of the PCOORD by the PCOORD, since at this time, the PCOORD is effectively the main controller.

B4: Once an error recovery has been completed, it is the responsibility of the PCOORD to update all relevant information to correctly reflect the status of the Intelligent Machine prior to returning control to the Intelligent Machine Main Controller.

In addition, once control reverts to the Intelligent Machine Main Controller, the PCOORD status reverts back to observer.

NOTE: Logically, the PCOORD is subservient to the Intelligent Machine Main Controller. At all times, even during an error recovery, the Intelligent Machine Main Controller can abort the PCOORD's operation. Only in the event of an Intelligent Machine Main Controller Failure does the PCOORD become system master, and even then only until the Main Controller is returned to an active state.

The second mode of operation for the PCOORD is on-line

planning. This mode is the subject of the following subsection.

4.3 On Line Path Planning

The on-line planning function of the Planning Coordinator can be viewed at times as a completely separate operation, distinct from error recovery, and at other times be viewed as a specific instantiation of an error recovery function. These two operational plans are referred to as:

i) Short Term On-Line Planning
ii) Interactive On-Line Planning

4.3.1 Short Term On-Line Planning

This type of on-line planning can be performed when the PCOORD is in an observer state and is the result of an insufficiency in the information available when the original off-line plan was made by the Organization Level of the Intelligent Machine. The additional information needed by the off-line planner is available after some sequence of events has taken place.

Effectively, the on-line planner takes advantage of the Planning Coordinator's error-recovery platform to create on-

line plans to be executed further on in the operational path of the Coordination Level Petri Net. This is accomplished by asserting the OP_FLAG by itself. Thus at an execution point where an on-line plan is known before hand to be needed, the on-line plan can have already been pre-generated (in a manner similar to that of creating the most likely errored event), utilizing the Current World Model, Primitive Structure Database and ERGA. To distinguish the on-line plan point in the Coordination Level Petri Net, a parallel point can be introduced and used as a trigger to the OP_FLAG.

If a plan is being pre-generated, it can be interrupted by a real error recovery. Hence error recovery takes precedence over on-line planning.

The second type of on-line planning incorporated into the error recovery platform of the Planning Coordinator, is interactive on-line planning, the subject of the next subsection.

4.3.2 Interactive On-Line Planning

When an on-line plan is needed, it is assumed that either the available information at the time of the original off-line

long-term planning stage was insufficient to characterize an efficient plan, or conditions have changed such that the previous off-line plan is no longer valid.

The former can be handled by the short term planner described in the previous subsection. The latter is the subject of this subsection. Note that although an error, in and of itself has not technically occurred, the operational parameters under which the task must run have changed such that continuing to utilize present parameters may or may not produce an error condition.

The original off-line plan must be capable of 'realizing' that such conditional changes are possible and, as such, allow for a special statement to verify that the expected conditions have not been altered. This special statement, executed at a 'node' or 'transition' in the operating Coordination Level Petri Net and SCPN is used as a trigger to the Planning Coordinator to examine the data and determine if an error is present. The trigger is the simultaneous assertion of the ERR_FLAG and OP_FLAG. If the parameter values are unaltered, the error recovery is aborted. If not, the choice to abort the error recovery or permit it to continue is made by the Intelligent Machine Main Controller. In this manner, the on-line planning is provided by the

recovery platform.

5.0 Summary and Continued Research

This paper has introduced the Planning Coordinator from a high-level perspective, including implementation detail only when clarification was warranted.

The Planning Coordinator as presented in this paper, provides a robust means of achieving both dynamic error recovery and on-line planning by combining the strengths of Petri Net synthesis techniques, high speed data storage and retrieval, and an adaptable combinatorial weighting scheme.

Research continues on the implementation aspects of the Planning Coordinator with preliminary research completed on the Mapping Mechanism, Primitive Structures, Primitive Structure Database, and Current World Model. Preliminary research continues on the Node/Link Weighting Mechanism and Error Recovery Generation Algorithm.

REFERENCES

[1] Saridis, G.N., "Architechtures For Intelligent Machines", CIRSSE #96, July 1991.

[2] Saridis, G.N. and Moed, M.C., "Analytic Formulation Of Intelligent Machines As Neural Nets", CIRSSE #1, .

[3] Saridis, G.N., "Intelligent Machines: Distributed Versus Hierarchical Intelligence", CIRSSE #2.

[4] Saridis, G.N., "An Analytic Formulation Of Knowledge Based Systems For Intelligent Machines", CIRSSE #3.

[5] Fielding, P., DiCesare, F. and Goldbogen, G., "Error Recovery in Automated Manufacturing through the Augmentation of Programmed Processes", Journal of Robotic Systems, 5(4), 337-362 (1988).

[6] Kelley, R.B. and Moed, M.C., "A Connectionist/Symbolic Model For Planning Robotic Tasks", CIRSSE #78, December 1990.

[7] Zhou, M.C. (1988), "Hybrid Synthesis Of Petri Nets For Manufacturing Systems", CIM Program Annual Meeting 1988.

[8] DiCesare, F. and Jeng, M.D. (1990), "A Review Of Synthesis Techniques For Petri Nets", CIM Rensselaer Polytechnic Institute.

[9] Narahari, Y. and Viswanadham, N. (1988), "Stochastic Petri Net Models For Performance Evaluation Of Automated Manufacturing Systems", Elsevier Science Publishers B.V. (North Holland).

[10] Koh, I. and DiCesare F., "Modular Transformation Methods For Generalized Petri Nets and Their Applications To Automated Manufacturing Systems", Rensselaer Polytechnic Institute.

Figures

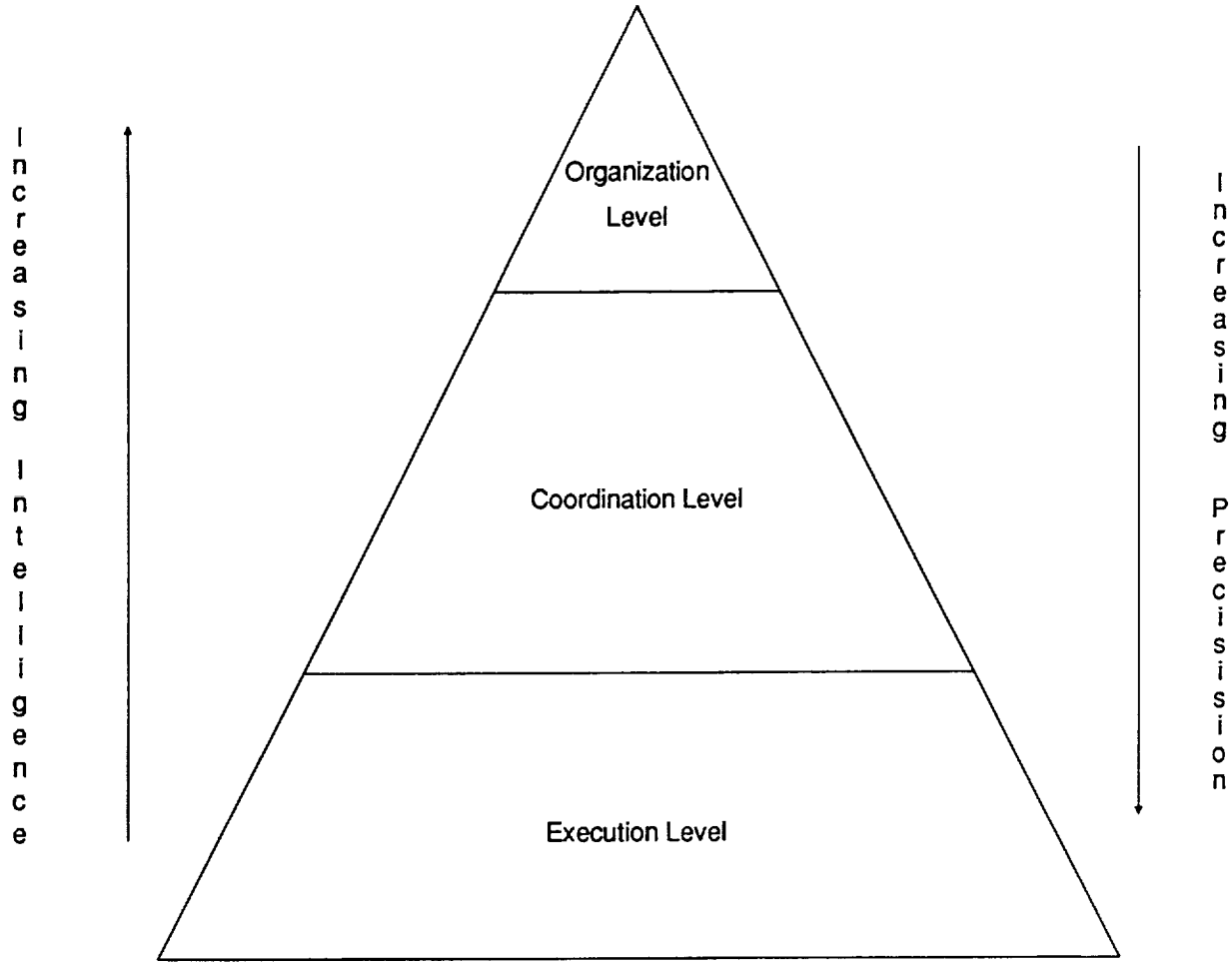

Figure 1

The Intelligent Machine Model [4]

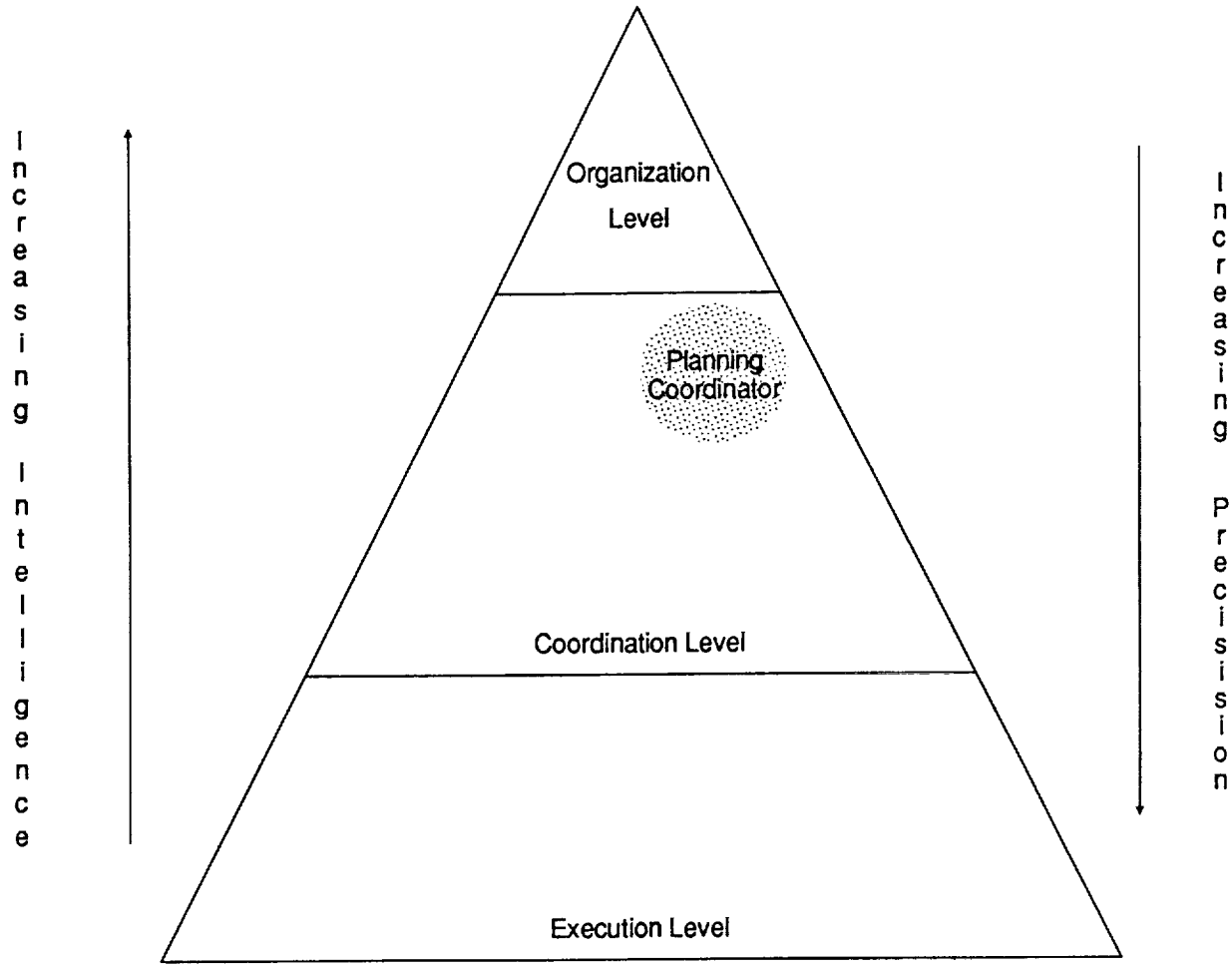

Figure 2

The Planning Coordinator - Logical Location
Within The Intelligent Machine Model Hierarchy of Saridis

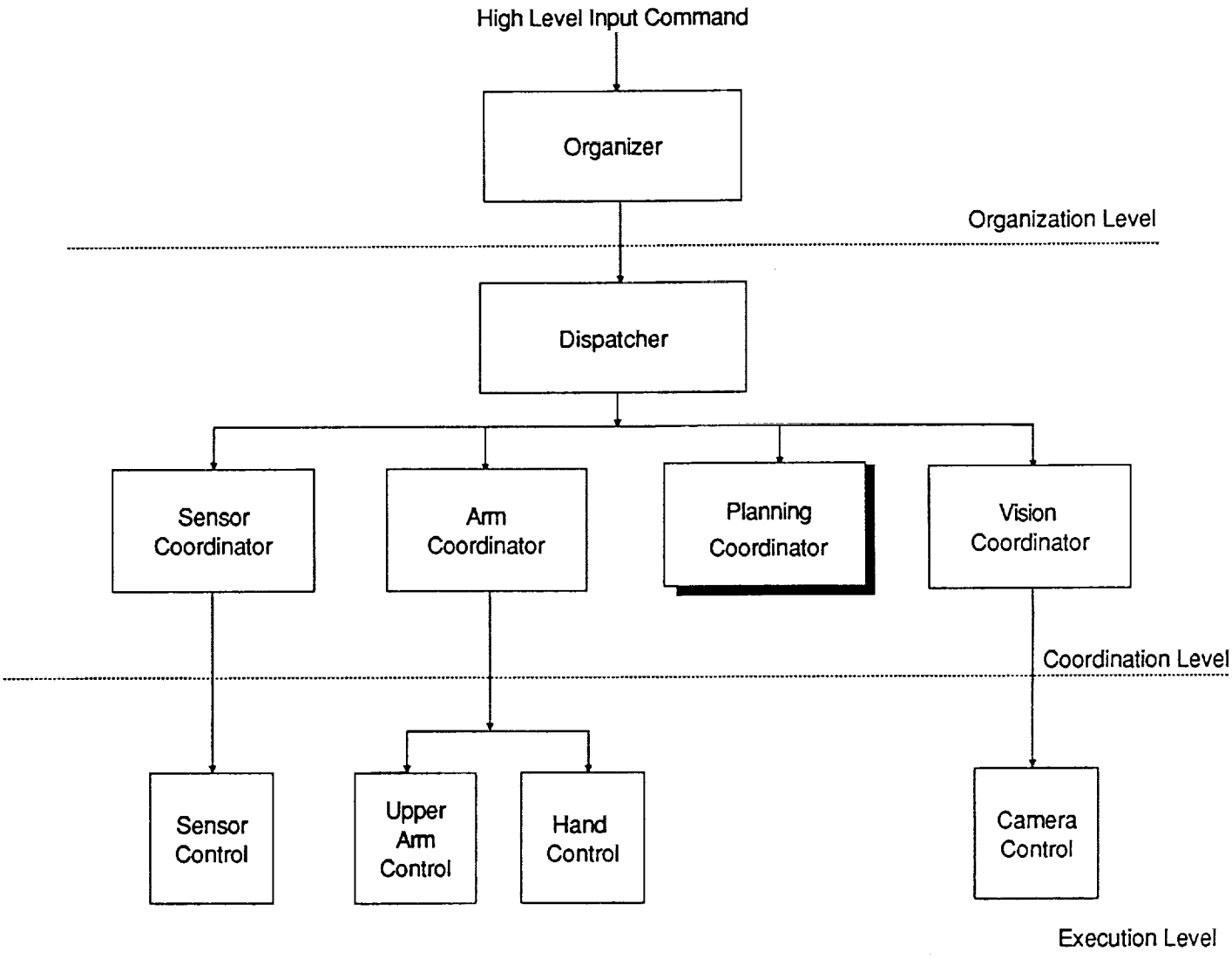

Figure 3

The Intelligent Machine Model Hierarchy
Functional Interconnection Block Diagram [1]

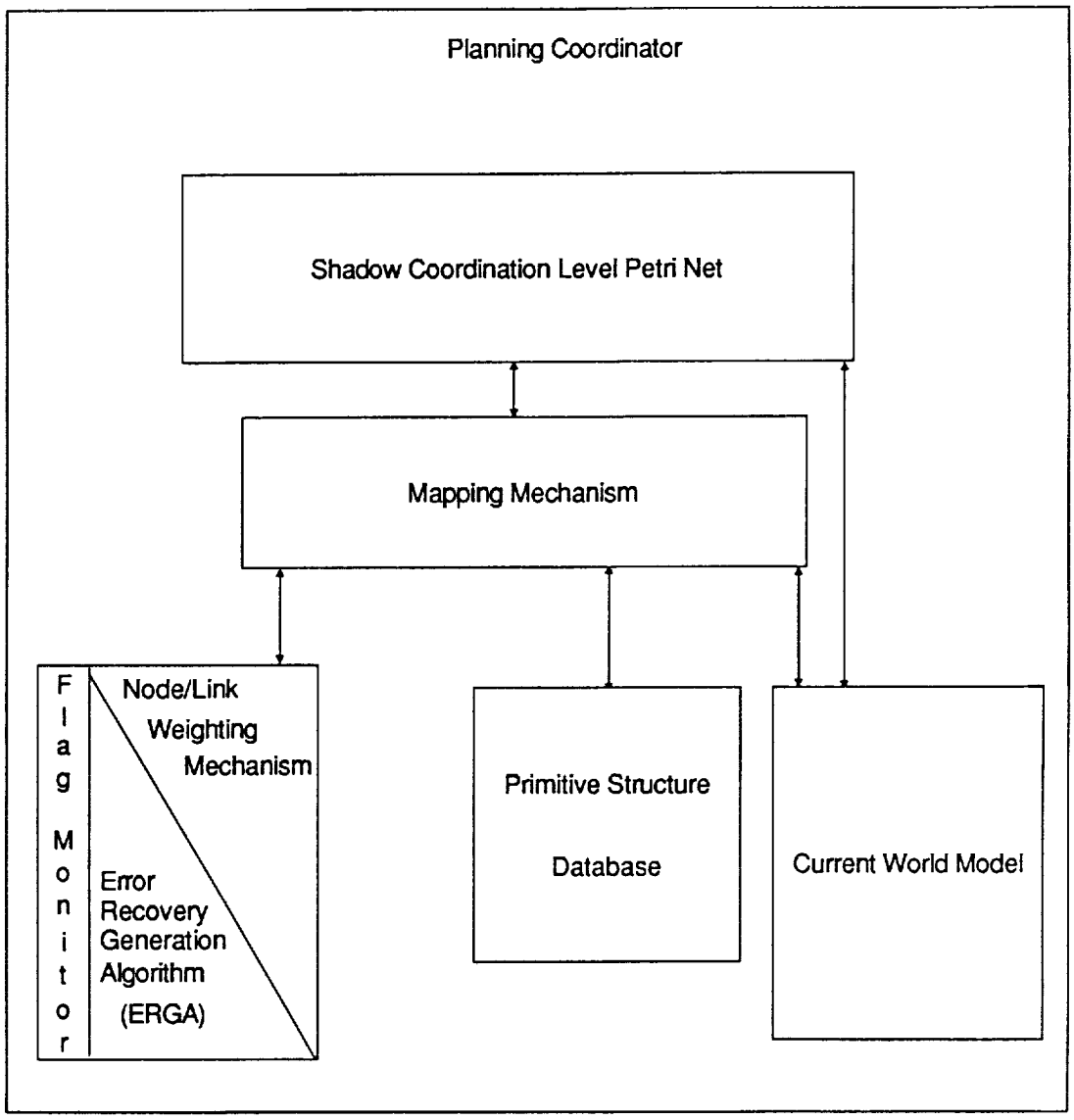

Figure 4

The Planning Coordinator (PCOORD)
Internal Functional Constituent Parts

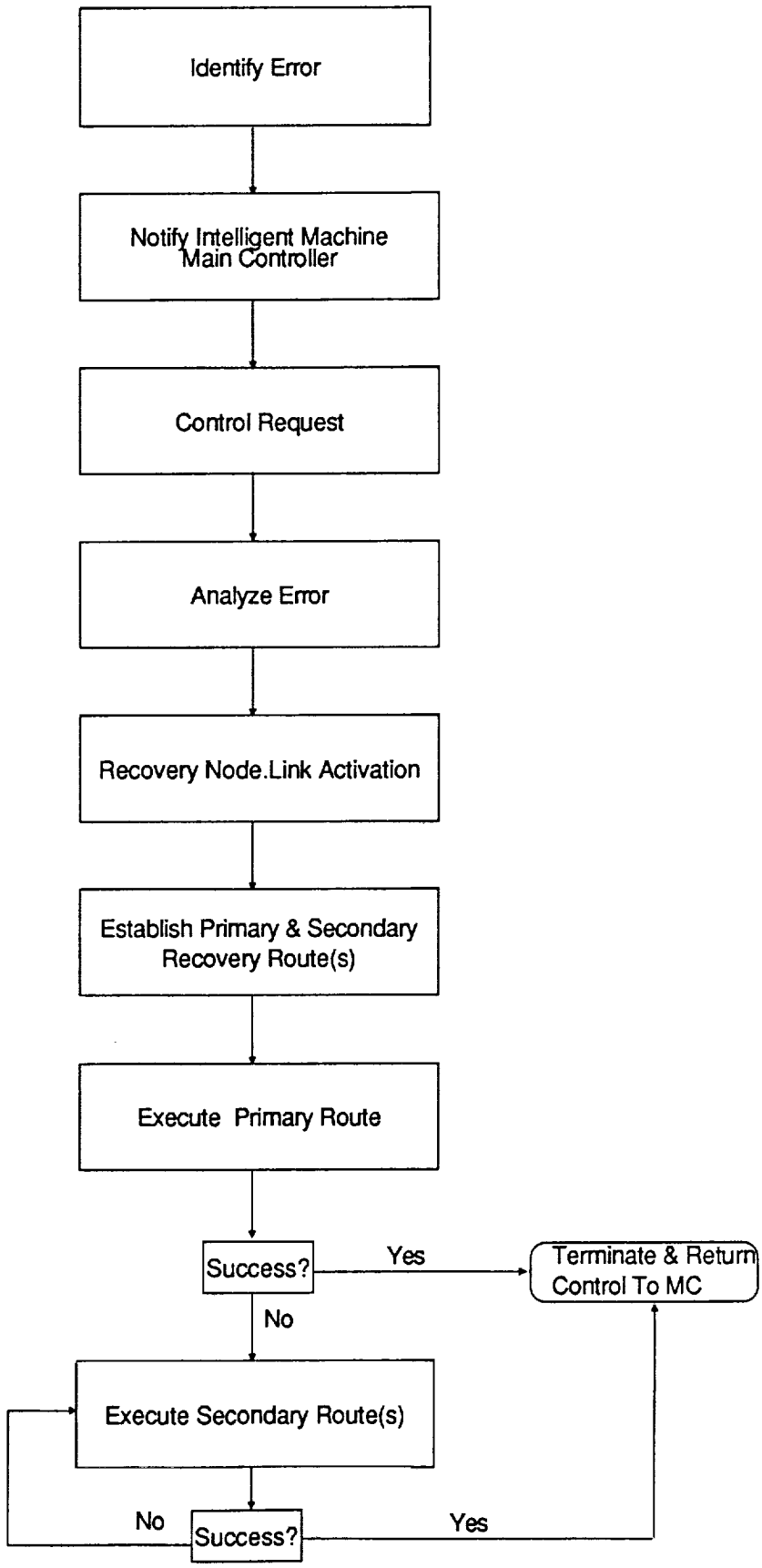

Figure 5
The Planning Coordinator (PCOORD)
Error Recovery Generation Algorithm (ERGA)

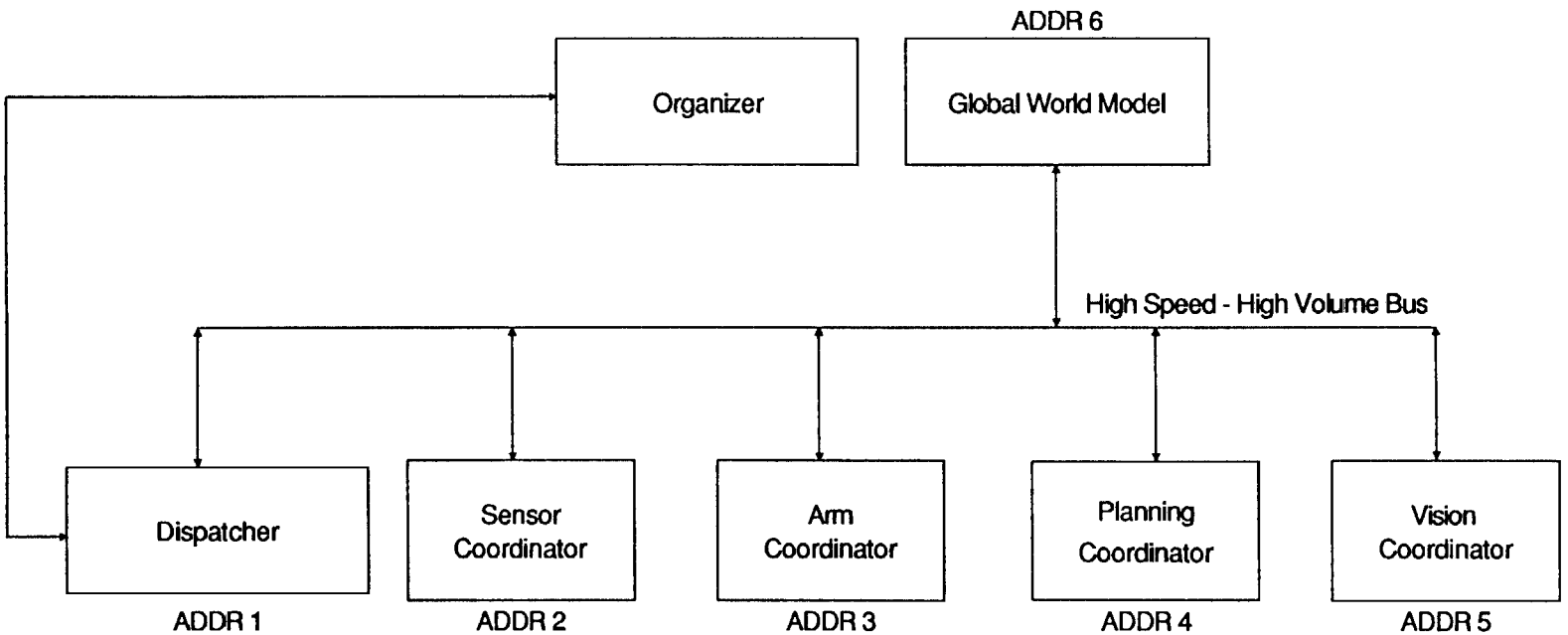

Figure 6

The Intelligent Machine Model Hierarchy
Communication Bus Architecture

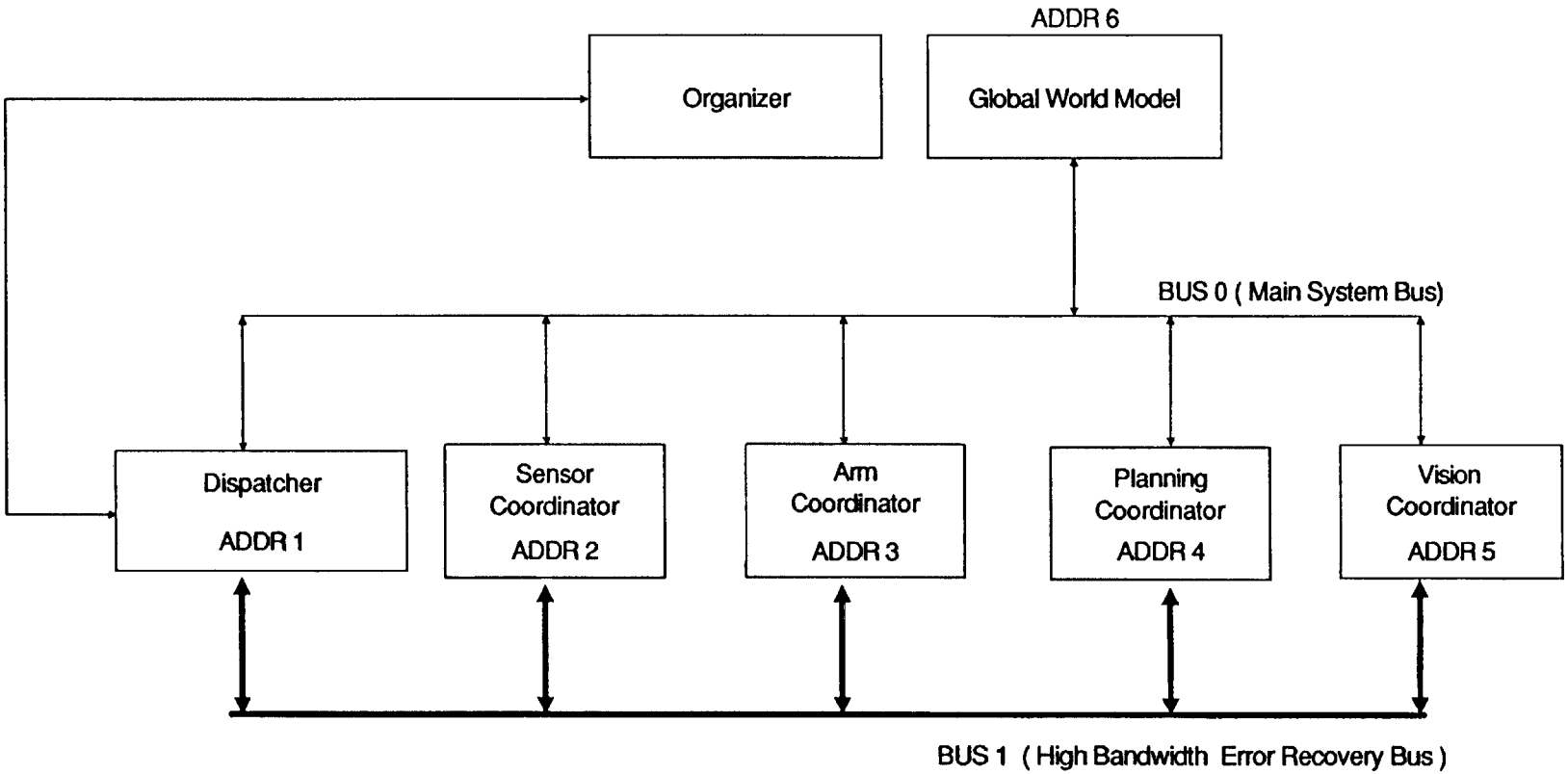

Figure 7

The Intelligent Machine Model Hierarchy
Communication Bus Architecture

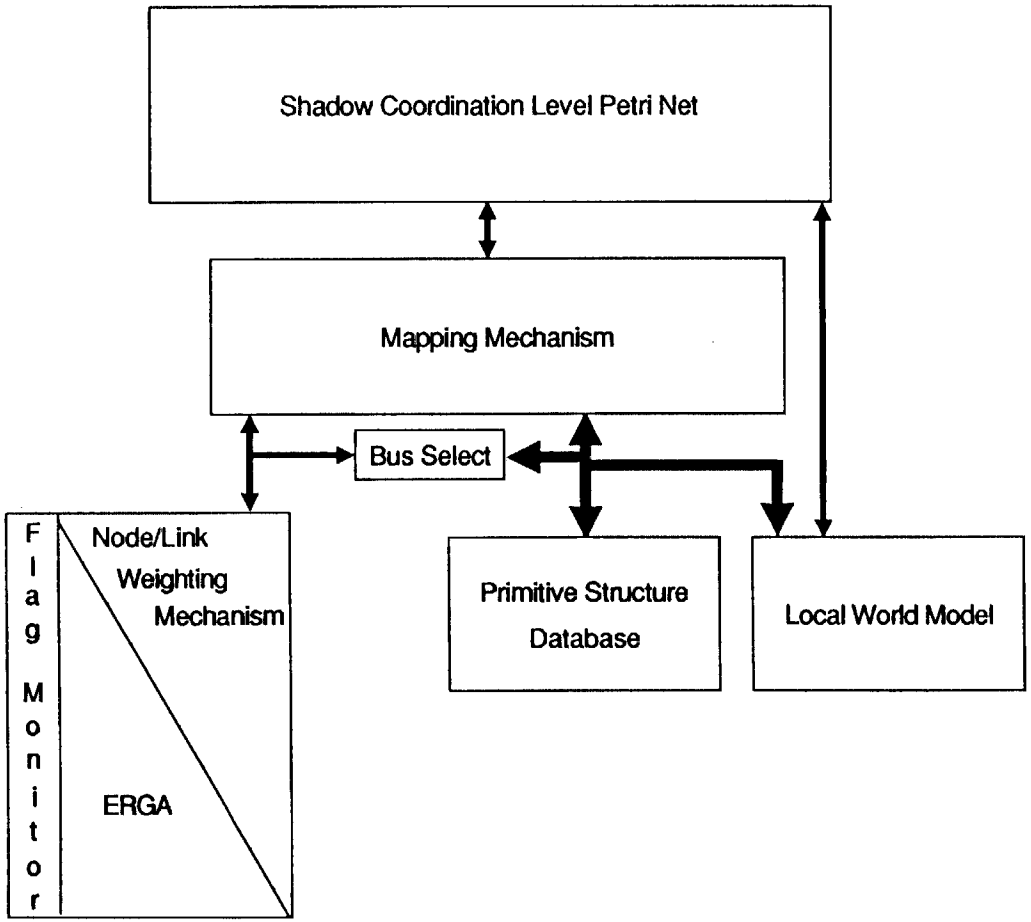

Figure 8

The Planning Coordinator (PCOORD)
Internal Communication Bus Structure

B1

State: Observer
ERR_FLAG = 0
Function:
Monitor System Operation
Perform other Functions

B2

State: Actor
ERR_FLAG = 1

If err_info = sufficient
{
PCOORD GeneratesPlan
PCOORD Requests System Control
If granted, goto B3 else system returns to B1
}

else
{
PCOORD Requests System Control
If granted, PCOORD Obtains Error Data else B1
PCOORD Generates Plan
PCOORD Requests Execution Clearance
If given, then B3
else B1
}

B3

State : Actor
ERR_FLAG = 1

PCOORD Sequentiall y Executes ERGA
Monitors Execution of Plan for New Errors
If New error, then B2 else continue

B4

State : Actor
ERR_FLAG=1

On completion of error recovery, PCOORD updates GWM and other sources with relevant information

PCOORD returns Intelligent Machine to success state (i.e. the position it expected to be in if error did not occur)

PCOORD requests clear of ERR_FLAG. If yes, then ERR_FLAG=0 & Status : Observer -> B1. Else B2

Figure 9
Block Diagram Description of PCOORD Operation
Error Recovery Mode